

MaskRay

- [Home](#)
- [Portfolio](#)
- [Presentations](#)
- [Subscribe](#)

[2013-08-27](#)

约瑟夫问题的两个 $O(\log n)$ 解法

这个是学习编程时的一个耳熟能详的问题了：

n 个人(编号为 $0, 1, \dots, n-1$)围成一个圈子，从 0 号开始依次报数，每数到第 m 个人，这个人就得自杀，之后从下个人开始继续报数，直到所有人都死亡为止。问最后一个死的人的编号(其实看到别人都死了之后最后剩下的人可以选择不自杀.....)。

这个问题一般有两种问法：

- 给出自杀顺序。不少数据结构初学书都会以这个问题为习题考验读者对线性表的掌握。比较常见的解法是把所有存活的人组织成一个循环链表，这样做时间复杂度是 $O(n*m)$ 的。另外可以使用order statistic tree(支持查询第 k 小的元素以及询问元素的排名)优化到 $O(n \log n)$ 。另外有篇1983年的论文*An $O(n \log m)$ Algorithm for the Josephus Problem*，但可惜我没找到下载链接。
- 求出最后一个人的编号。可以套用上一种问法的解法，但另外有更加高效的解法，下文展开讨论。

时间 $O(n)$ ，空间 $O(1)$

设 $f(n)$ 为初始有 n 人时最后一个自杀的人的编号，那么有如下递推式：

$$f(n) = (f(n-1) + m) \bmod n$$

以 $n=5, m=3$ 为例，一开始有这么5个人：

1 0 1 2 3 4

第一轮报数后2号死亡，圈子里剩下来的人的编号是这些：

1 3 4 0 1

这里把3号写在最前面了，因为轮到3开始报数。如果有办法知道 $n=4$ 时的答案，即初始圈子为：

1 0 1 2 3

时的答案，那么可以把 $n=4$ 的初始圈子的所有数 x 变换成 $(x+3) \bmod 5$ ，得到：

1 3 4 0 1

这个恰好就是 $n=5$ 时第一轮结束后的圈子状态，也就是说我们可以根据 $n=4$ 的答案推出 $n=5$ 时的答案。

手工演算一下就能发现 $n=z$ 时的圈子第一轮结束后(即 $m-1$ 号自杀后)的圈子状态，可以由 $n=z-1$ 的初始圈子施以变换 $x \rightarrow (x+m) \bmod z$ 得到。于是我们可以从 $n=1$ 开始(此时的答案是0)，推出 $n=2$ 的答案，再推出 $n=3$ ，直到计算到所要求的 n 。

下面是C语言实现：

```
1 int f(int n, int m)
2 {
3     int s = 0;
4     for (int i = 2; i <= n; i++)
5         s = (s + m) % i;
6     return s;
7 }
```

时间 $O(\log n)$ ，空间 $O(\log n)$

换一个递推思路，考虑报数过程又回到第0个人时会发生什么。这时有 $\text{floor}(n/m) * m$ 个人都已经报过数了，并且死了 $\text{floor}(n/m)$ 人。之后一轮的报数会数过 m 个人，又会回到第0个人。

我们以 $n=8$ ， $m=3$ 为例看看会发生什么。一开始：

1 0 1 2 3 4 5 6 7

$\text{floor}(n/3) * 3$ 个人都报过数后：

1 0 1 x 3 4 x 6 7 (A)

即2号和5号死亡，接下来轮到6号开始报数。因为还剩下6个人，我们设法做一个变换把编号都转换到0~5：

1 2 3 x 4 5 x 0 1 (B)
2 —

(B)的答案等于规模缩小后的情况 $n'=6$ 时最后一个人的编号 s 。然后根据 s 算出圈子状态(A)的最后一个人的编号。

如果 s 在最后一个 x 后面(带下划线)，即 $s < n\%m$ ，那么 $s_2 = s - n\%m + n$ ；否则 $s_2 = s - n\%m + (s - n\%m) / (m - 1)$ ；

注意如果 $n < m$ ，那么报数回到第一个人时还没死过人。因为子问题的规模没有减小，所以上述方法不适用。需要用之前时间复杂度 $O(n)$ 的方法递推。下面是C语言实现：

```
1 int rec(int n, int m)
2 {
3     if (n == 1) return 0;
4     if (n < m) return (rec(n - 1, m) + m) % n;
5     int s = rec(n - n / m, m) - n % m;
6     return s < 0 ? s + n : s + s / (m - 1);
7 }
```

n 每次变为 $n * (m - 1) / m$ ，即以一个常数因子衰减到刚好小于 m ，然后换用线性复杂度的递推算法，总的时间复杂度为 $O(m + \log_{\{m/(m-1)\}}\{n/m\})$ ，如果把 m 看作常数的话就是 $O(\log n)$ 。程序中在子问题计算后还需要做一些处理，无法尾递归，所以空间复杂度也等于这个。

参考：

- Time Improvement on Josephus Problem (2002) by Fatih Gelgi
- <http://stackoverflow.com/questions/4845260/josephus-for-large-n-facebook-hacker-cup>

时间 $O(\log n)$ ，空间 $O(1)$

三年前我还看到过一段更巧妙的代码，具体写法不可考了，当时盯着式子思考了好久呢。其等价的形式长这样：

```
1 int kth(int n, int m, int k)
2 {
3     if (m == 1) return n - 1;
4     for (k = k * m + m - 1; k >= n; k = k - n + (k - n) / (m - 1));
```

```

5   return k;
6 }

```

这个函数解决了第二种问法的一个扩展问题，即第 k 个(从0数起)自杀的人的编号。如果取 $k = n-1$ 那么就得到了最后一个自杀的人的编号。

这个算法的思想是第 $k*m+m-1$ 次报数的人就是第 k 个自杀的人，追踪他之前每次报数的时刻来确定他的编号。考虑 $n=5, m=3, k=5$ ，一共有15次报数(每报 m 次数死一个人，一共有 $k+1$ 个人要死，所以需要 $(k+1)*m$ 次报数)，每一次报数的人的编号如下：

```

1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
2 0 1 2 3 4 0 1 3 4 1 3 1 3 3

```

报到2、5、8、11、14的人自杀。

设第 p 次(从0数起)报数的人是 y ，令 $p = m*a+b$ ($0 \leq b < m$)。经过前 p 次报数，一共死了 $\text{floor}(p/m) = a$ 人，圈子里还剩下 $n-a$ 人。如果 y 本次报数后没有自杀，那么他下次报数是在圈子里其他 $n-a$ 人都报数之后，也就是第 $q = p+n-a = n+(m-1)*a+b$ 次报数。这是后继的计算式，逆用即可计算前驱。

由 y 本次报数后还存活知 $b < m-1$ ，故 $a = \text{floor}((q-n)/(m-1))$ ， $p = q-(n-a) = q-n+\text{floor}((q-n)/(m-1))$ 。

我们要求第 k 个自杀的人，他是第 $k*m-1$ 次报数后自杀的，用计算前驱的方式求出这个人之前一次报数是在什么时候，不断地重复这一过程，直到知道他是第 k' ($0 \leq k' < n$)次报数的人，那么 k' 就是这个人的编号。

[algorithm](#), [josephus problem](#)

Comments

0 Comments MaskRay's Blog

1 Login ▾

♥ Recommend ↗ Share

Sort by Best ▾



Start the discussion...

Be the first to comment.

ALSO ON MASKRAY'S BLOG

BCTF 2016 hsab及BetaFour命题报告

1 comment • 8 months ago •



lowkey — nice trick!

从ASC'14到ISC'15——我的超算竞赛生涯谢幕

11 comments • a year ago •



BYVoid — 我去打酱油竟然也被宋教授提及了。

DEFCON 24 CTF参赛记

2 comments • 3 months ago •



laike9m — 膜拜教授雄文

支持iOS 9内置IKEv2 VPN客户端的strongSwan 5.3.5配置

2 comments • a year ago •



meng — http://task.zbj.com/7770338

✉ Subscribe ⓘ Add Disqus to your site Add Disqus Add 🔒 Privacy

Popular

- [Android微信数据导出 \(6586\)](#)
- [完美迷宫生成算法 \(6195\)](#)
- [Evil--在Emacs中模拟Vim \(5821\)](#)
- [LeetCode solutions \(4871\)](#)
- [约瑟夫问题的两个 \$O\(\log n\)\$ 解法 \(4680\)](#)
- [支持Android、iOS 9内置IPSe... \(4154\)](#)
- [使用Suricata进行IDS/IPS \(3781\)](#)
- [《Debug Hacks》和调试技巧 \(3571\)](#)
- [LeetCode Best Time to... \(3196\)](#)
- [2014年总结——竞赛的一年 \(2774\)](#)
- [Force-directed算法\(1\)——... \(2560\)](#)
- [DEFCON 21 CTF参赛记 \(2484\)](#)
- [从ASC'14到ISC'15——我的超算竞... \(2355\)](#)
- [我的xmonad配置 \(2310\)](#)

- [RedTigers Hackit Warg... \(2119\)](#)
- [Python is ugly \(1978\)](#)
- [DEFCON 22 CTF参赛记 \(1917\)](#)
- [DEFCON 23 CTF参赛记 \(1794\)](#)
- [BCTF工作组记事 \(1780\)](#)
- [OverTheWire - Natas W... \(1705\)](#)
- [皈依Emacs \(1476\)](#)
- [BCTF 2015 CamlMaze命题报... \(1458\)](#)
- [8门编程语言的设计思考 \(1234\)](#)
- [DEFCON 24 CTF参赛记 \(1207\)](#)
- [建立清华大学Node Packaged M... \(1151\)](#)
- [wechatircd——用IRC客户端控制... \(1057\)](#)
- [调试技巧2 \(990\)](#)
- [SVT13117ECS上Gentoo安装记... \(899\)](#)
- [HDU OJ一些题目 \(870\)](#)
- [80分钟8语言 \(843\)](#)
- [用Google Analytics API... \(816\)](#)
- [一次服务器BMC固件逆向经历 \(801\)](#)
- [D-Link路由器后门注记 \(791\)](#)
- [J语言初探 \(727\)](#)
- [DEFCON 21——我的奋斗 \(704\)](#)
- [BCTF 2016 hsab及BetaFo... \(620\)](#)
- [并行N-body模拟 \(610\)](#)
- [Trend Micro Codinsani... \(587\)](#)
- [ML编译器Caml Featherweig... \(572\)](#)
- [指定dynamic linker以使用高版... \(568\)](#)
- [Haskell学习笔记 \(549\)](#)
- [自然语言处理之词语抽取 \(548\)](#)
- [用DocPad构建静态网站 \(544\)](#)
- [二叉树遍历算法总结 \(526\)](#)
- [“1001夜”学生节线上解谜活动1001/... \(516\)](#)
- [终端模拟器下使用双倍宽度多色Emoji字体 \(507\)](#)
- [Ouroborus Program - q... \(504\)](#)
- [在Makefile中自动生成依赖 \(493\)](#)
- [LeetCode Expression A... \(487\)](#)
- [TUNA技术沙龙及A Pretty Pri... \(486\)](#)
- [String index structur... \(481\)](#)
- [第二届青少年开发者大会 \(475\)](#)
- [RuCTFE 2012参赛记 \(460\)](#)
- [江苏信安竞赛初赛工作组记事 \(450\)](#)
- [异于Make的另一种构建系统 \(411\)](#)

- [Ko-Aluru suffix array... \(400\)](#)
- [基于 mutt+offlineimap+n... \(394\)](#)
- [ML编译器Caml Featherweig... \(391\)](#)
- [ISC'14 Student Cluste... \(384\)](#)
- [ELF Hacks \(380\)](#)
- [使用Burkhard-Keller tre... \(364\)](#)
- [SECUINSIDE CTF Quals ... \(363\)](#)
- [gcc中sqrt实现 \(356\)](#)
- [用udev自动挂载usb设备 \(337\)](#)
- [用Makefile搭建博客 \(329\)](#)
- [偃师——finite automaton生成工具 \(329\)](#)
- [自动获取SSH密码并登录 \(304\)](#)
- [使用Double-array trie优化... \(302\)](#)
- [无需每日扫码的IRC版微信和QQ：wech... \(276\)](#)
- [Haskell实现的Splay树 \(258\)](#)
- [Cyber Grand Challenge... \(254\)](#)
- [jq实现原理——字节码 \(252\)](#)
- [常数空间Invert Binary Tre... \(247\)](#)
- [智能体大赛与平台开发 \(231\)](#)
- [Js_of_ocaml和Emscripten \(228\)](#)
- [Makefile介绍 \(213\)](#)
- [用Perl的正则表达式分析csv文件 \(209\)](#)
- [Nginx根据Accept-Languag... \(208\)](#)
- [脱离chroot的枷锁 \(195\)](#)
- [用Makefile搭建博客-2 \(189\)](#)
- [WeeChat操作各种聊天软件 \(183\)](#)
- [用Pike's VM实现的非回溯正则表达式引擎 \(177\)](#)
- [telegramircd——用IRC客户端... \(177\)](#)
- [Gmail的OfflineIMAP XOA... \(159\)](#)
- [计算所有后缀的排名 \(153\)](#)
- [webqqircd——用IRC客户端控制W... \(149\)](#)
- [三柱汉诺塔相关扩展问题 \(144\)](#)
- [PyGTK迷宫生成器mazer \(139\)](#)
- [八数码 \(134\)](#)
- [GTK+黑白棋reversi \(131\)](#)
- [域名迁移至maskray.me \(131\)](#)
- [整系数Discrete Fourier t... \(124\)](#)
- [NOIP 2004 数字游戏（虫食算） \(116\)](#)
- [用Expect连接无线网络 \(116\)](#)
- [终端模拟器下宽字符退格 \(116\)](#)
- [网络学堂feeds2mail \(112\)](#)
- [Tiling window manager... \(103\)](#)

- [根据计划求解Rush \(86\)](#)
- [GTK+ tic-tac-toe（井字棋） \(81\)](#)
- [用rss2email阅读feeds \(80\)](#)
- [称球问题扩展——根据当前已知信息选择最优称量方案 \(72\)](#)
- [给xbindkeys添加key seque... \(71\)](#)
- [genkernel \(60\)](#)
- [USACO JAN10 Gold \(54\)](#)
- [ZJU 1638. Greedy Island \(51\)](#)
- [Dancing Links+Algorit... \(42\)](#)
- [Awesome 3 debian menu \(40\)](#)
- [.emacs \(38\)](#)
- [FFT \(38\)](#)
- [XV Olimpiada Informat... \(28\)](#)
- [checker \(13\)](#)

Tag Cloud

[adc](#) [ai9](#) [algorithm](#) [asc](#) [automaton](#) [awesome](#) [bctf](#) [binary](#) [bmc](#) [build system](#) [c](#) [c++](#) [cgc](#) [chroot](#) [codinsanity](#) [coffee script](#) [compiler](#) [computer](#)
[security](#) [contest](#) [csv](#) [ctf](#) [data structure](#) [debug](#) [defcon](#) [desktop](#) [docker](#) [elf](#) [emacs](#) [email](#) [emoji](#) [emscripten](#) [event](#) [expect](#) [feeds](#)
[firmware](#) [floating point](#) [forensics](#) [game](#) [gcc](#) [gentoo](#) [graph drawing](#) [gtk](#) [hanoi](#) [haskell](#) [hpc](#) [inotify](#) [ipsec](#) [irc](#) [isc](#) [j](#) [javascript](#) [josephus problem](#) [jq](#)
[kernel](#) [ld](#) [leetcode](#) [linux](#) [makefile](#) [math](#) [maze](#) [mirror](#) [ml](#) [mutt](#) [n-body](#) [network](#) [nginx](#) [nlp](#) [node.js](#) [noip](#) [notmuch](#) [npm](#) [ocaml](#) [offlineimap](#) [oi](#)
[oj](#) [parallel](#) [parser generator](#) [perl](#) [presentation](#) [puzzle](#) [python](#) [qq](#) [regex](#) [regular expression](#) [reverse engineering](#) [review](#) [ruby](#) [ructfe](#) [scheme](#)
[search](#) [security](#) [shell](#) [ssh](#) [stringology](#) [student festival puzzle](#) [suffix array](#) [suffix automaton](#) [summary](#) [suricata](#) [telegram](#) [telegramircd](#) [terminal](#)
[traversal](#) [tree](#) [trendmicro](#) [udev](#) [unicode](#) [usb](#) [vim](#) [vpn](#) [vte](#) [wargame](#) [web analytics](#) [webqqircd](#) [website](#) [wechat](#) [wechatircd](#) [window manager](#)
[xbindkeys](#) [xmonad](#) [yanshi](#)

Blogroll

- [BYVoid](#)
- [fqj1994](#)
- [ppwwyyxx](#)

© 2016 MaskRay